OBJECT-ORIENTED PROGRAMMING
HY-252

George Kokolakis

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

INTRODUCTION

In this project we use, the model MVC
(Model,view,controller).
The model package will be the kernel of the application, the view will contain the graphics and the interaction between user and Board, and finally the controller will connect the model with the view, in order to get the final result. Down below I will analyze what the three packages contains and how they work.

Package Model

This package contain ,the Abstract class Card , the classes NumberCard and SorryCard which inherit the class Card.
Also
it contains the abstract class Square and it's subclasses StartSquare , SimpleSquare ,SlideSquare , SafetyZoneSquare and HomeSquare. In the end we have the classes Deck , MoveChecker ,Pawn and Player.

Class Card And Subclasses

Abstract Class Card

Variables:

```
Boolean has_been_played; //This variable check's if the
current card has been played
private String image; //Path of the image
private String name; //Name of the card
private String [] choices;//the choices that the player can make
for moving the pawn according the current card
```

Methods:

1. public Card(String image , String name) (constructor)

Saves the path of the image and the name of the card

2. public String getName(); (accessor)

Gets the name of the current card

3. public void setName(String name); (Transformer)

Set the name of the card

4. public Boolean has_been_played(); (Observer)

Check if the card has already played once

5. public void sethas_been_played(Boolean has_been_played);   (Transformer)

Set if the current card has been played

6. public abstract void movePawn(Pawn pawn,Deck Deck);

This method moves the pawn depending on the card

7. public abstract String toString();

This method returns the representation of the card as a string

8. public String getImage(); (Accessor)

Get the path of the image for the current card

9. public void setImage(String image); (Transformer)

Set the path for the image of the current card

10.public String[] getChoices() (Accessor)
get the choices that the current card gives to user to make.

11.public void update_Pawn_Current_Square(Pawn pawn , int square_Index ,   Deck deck)

This method updates the squares current pawn after the move was made

12.public void set_Squares_Pawn(Pawn pawn , int square_Index , Deck deck)

This method updates the pawn current square after it moves

Class NumberCard And SorryCard

Class NumberCard:

Variables:

private int value;//card value

Methods:

1. public NumberCard(String image, String name,int value);
(Constructor)

sets the name and the path of the image for the card through
super constructor and sets the card's value

2. public void setValue(int value) (Transformer)

Set the value of the current card

3. public String toString();

return the String representation for the current card

4. public int getValue(); (Accessor)

return the value of the current card

5. public void movePawn(Pawn pawn, Deck deck);

This method Moves the pawn

Class SorryCard:

Variables:

private Pawn enemyPawn;//Enemy pawn

Methods:

1. public void setEnemyPawn(Pawn enemyPawn);
(Transformer)

Sets the enemy pawn

2. public SorryCard(String image, String name) (constructor)

Initialize the path of the image and the name of the card through super constructor

3. public void movePawn(Pawn pawn, Deck deck)

This method moves the pawn following the rules of the sorry card

4. public void GetEnemyPawn() (Accessor)
Get the enemy's pawn

5. public String toString()

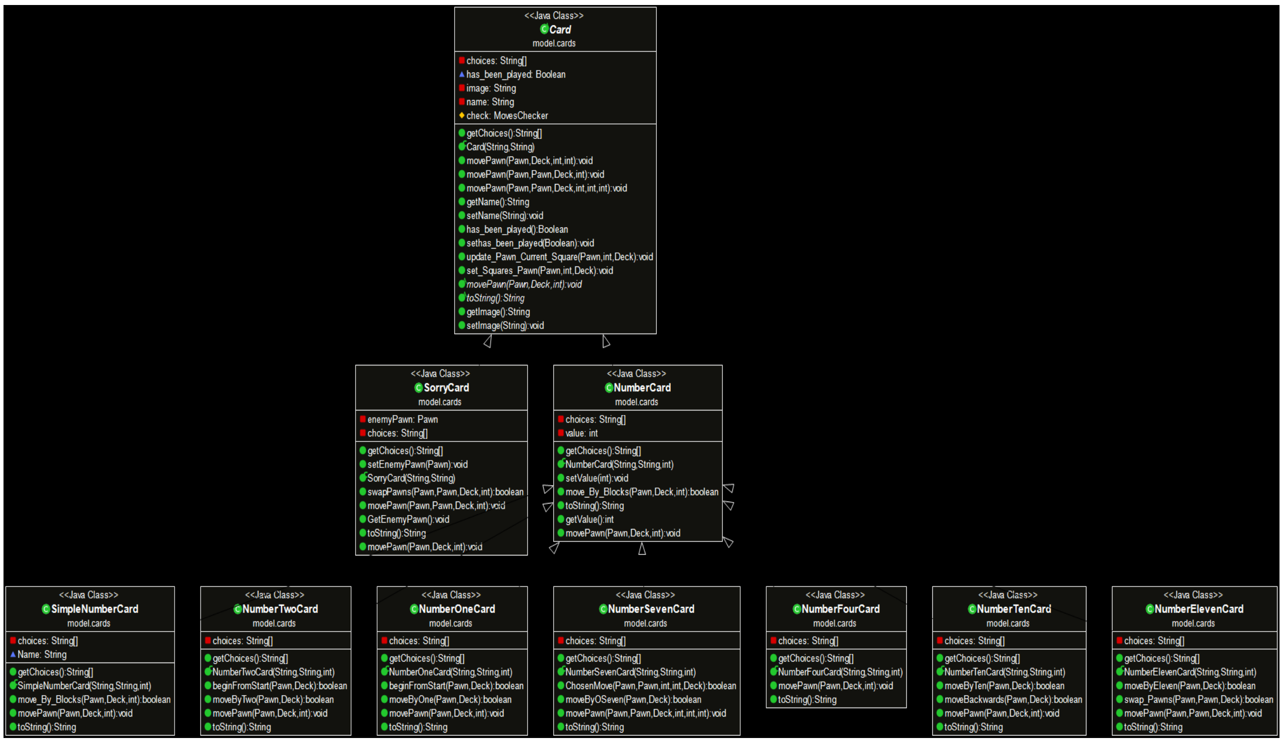
string representation of the sorry card

Other Card Classes :

The rest of the card classes , SimpleNumberCard,NumberOneCard. NumberTwoCard,NumberFourCard,NumberSevenCard,NumberTenCard ,NumberElevenCard are subclasses of the NumberCard.

All of them call the super constructor and implement a movePawn() and a toString() method.

Card Classes UML Presentation



<<Java Class>>
Card
model.cards
choices: String[]
has_been_played: Boolean
image: String
name: String
check: MovesChecker
getChoices():String[]
Card(String,String)
movePawn(Pawn,Deck,int,int):void
movePawn(Pawn,Pawn,Deck,int):void
movePawn(Pawn,Pawn,Deck,int,int,int):void
getName():String
setName(String):void
has_been_played():Boolean
sethas_been_played(Boolean):void
update_Pawn_Current_Square(Pawn,int,Deck):void
set_Squares_Pawn(Pawn,int,Deck):void
movePawn(Pawn,Deck,int):void
toString():String
getImage():String
setImage(String):void
<<Java Class>>
SorryCard
model.cards
enemyPawn: Pawn
choices: String[]
getChoices():String[]
setEnemyPawn(Pawn):void
SorryCard(String,String)
swapPawns(Pawn,Pawn,Deck,int):boolean
movePawn(Pawn,Pawn,Deck,int):void
GetEnemyPawn():void
toString():String
movePawn(Pawn,Deck,int):void
<<Java Class>>
NumberCard
model.cards
choices: String[]
value: int
getChoices():String[]
NumberCard(String,String,int)
setValue(int):void
move_By_Blocks(Pawn,Deck,int):boolean
toString():String
getValue():int
movePawn(Pawn,Deck,int):void
<<Java Class>>
SimpleNumberCard
model.cards
choices: String[]
Name: String
getChoices():String[]
SimpleNumberCard(String,String,int)
move_By_Blocks(Pawn,Deck,int):boolean
movePawn(Pawn,Deck,int):void
toString():String
<<Java Class>>
NumberTwoCard
model.cards
choices: String[]
getChoices():String[]
NumberTwoCard(String,String,int)
beginFromStart(Pawn,Deck):boolean
moveByTwo(Pawn,Deck):boolean
movePawn(Pawn,Deck,int):void
toString():String
<<Java Class>>
NumberOneCard
model.cards
choices: String[]
getChoices():String[]
NumberOneCard(String,String,int)
beginFromStart(Pawn,Deck):boolean
moveByOne(Pawn,Deck):boolean
movePawn(Pawn,Deck,int):void
toString():String
<<Java Class>>
NumberSevenCard
model.cards
choices: String[]
getChoices():String[]
NumberSevenCard(String,String,int)
ChosenMove(Pawn,Pawn,int,int,Deck):boolean
moveByOSeven(Pawn,Deck):boolean
movePawn(Pawn,Pawn,Deck,int,int,int):void
toString():String
<<Java Class>>
NumberFourCard
model.cards
choices: String[]
getChoices():String[]
NumberFourCard(String,String,int)
movePawn(Pawn,Deck,int):void
toString():String
<<Java Class>>
NumberTenCard
model.cards
choices: String[]
getChoices():String[]
NumberTenCard(String,String,int)
moveByTen(Pawn,Deck):boolean
moveBackwards(Pawn,Deck):boolean
movePawn(Pawn,Deck,int):void
toString():String
<<Java Class>>
NumberElevenCard
model.cards
choices: String[]
getChoices():String[]
NumberElevenCard(String,String,int)
moveByEleven(Pawn,Deck):boolean
swap_Pawns(Pawn,Pawn,Deck):boolean
movePawn(Pawn,Pawn,Deck,int):void
toString():String

Class Square And Subclasses

Abstract Class Square:

Variables:

```
private Color color;   //color of the current square

private boolean hasPawn; //variable for checking if
the square has pawn

private Pawn curr_pawn; //variable for getting the pawn
that is in the square

private String image ;//path of the image for the current
```

square

Methods:

1. public void setCurr_pawn(Pawn curr_pawn); (Transformer)

This method set the pawn to the current square

2. public Pawn getCurr_pawn(); (Accessor)

This method gives us the pawn on the current square

3. public Square(Color color,String image); (constructor)

set the color of square and the image path

4. public String getImage(); (Accessor)

return The path of the image for the current square

5. public void setImage(String image) (Transformer)

Set the image path for the current square

6. public boolean hasPawn(); (Observer)

This method check's if the current square has a pawn on it

7. public void setColor(Color color); (Transformer)

Set the color for the current square

8. public Color getColor(); (Accessor)

```
    return the color of the current square
```

Other Square Classes :

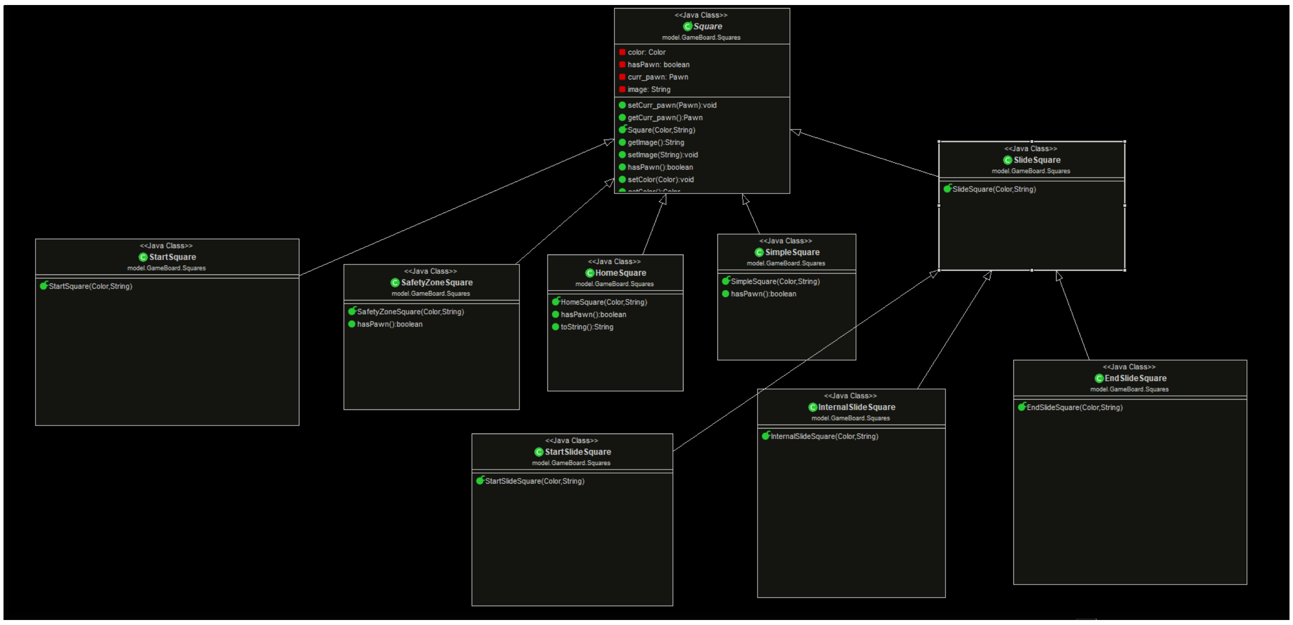
The rest of the square classes,StartSquare,SimpleSquare,SlideSquare, SafetyZoneSquare,HomeSquare are subclasses of the class Square.

```
All of them call the Super Constructor for initializing their
color and their path image.
```

The classes : StartSlideSquare , InternalSlideSquare , EndSlideSquare are subclasses of the class SlideSquare.

```
All of them call the Super Constructor for initializing their
color and their path image.
```

Square Classes UML Presentation



Class Pawn

Variables:

```
private Color color;//color of the pawn

private Square currentSquare;//square that the pawn is on

private boolean available;//if the pawn is available ,so
if the pawn reached the end or not

private int position;//the position of the current pawn
```

Methods:

1. public Color getColor(); (Accessor)

This method Gives us the color of the current pawn

2. public int getPosition(); (Accessor)

This method returns the position of the current pawn

3. public void setPosition(int position); (Transformer)

This method set the position of the current pawn

4. public boolean isAvailable(); (Accessor)

his method gives us the information if the current pawn has reached the end

5. public Square getCurrentSquare(); (Accessor)
This method is used for recognizing the square that the pawn stands

Class MoveChecker

This class contain methods that will be used in the movePawn() method. Those methods are responsible for checking certain special cases which doesn't allow the pawn to move.

Methods:

1. public boolean IsEnemyStartSquare(Pawn pawn , int pos,Deck deck) (Observer)

This method checks if the move that the user chose , will lead to move the pawn to Enemy start square

2. public boolean IsEnemySafetyZone(Pawn pawn , int pos , Deck deck) (Observer)

This method checks if the move that the user chose , will lead to move the pawn to Enemy safety zone square

3. public boolean IsEnemyHome(Pawn pawn , int pos , Deck deck) (Observer)

This method checks if the move that the user chose , will lead to move the pawn to Enemy Home square

4. public boolean IsTwoSamePawnsOnSquare(Pawn pawn , int pos , Deck deck) (Observer)

This Method checks if the move that the user chose will lead to move the pawn to a square that there is already a friendly pawn

5. public boolean IsEnemyPawnOnSquare(Pawn pawn , int pos,Deck deck) (Observer)

This Method checks if the move that the user chose , will lead to move the pawn to a square that there is an enemy pawn

Class Player

Variables:

```
private Color color;//color of the player
private String name;//name of the player
private Pawn pawn1;//his first pawn
private Pawn pawn2;//his  second pawn
private boolean canPlay;//if it's his turn or not
private boolean finished;//if the player has finished the
game
```

Methods:

1. `public Player(String name , Pawn pawn1,Pawn pawn2);` (Constructor)

This constructor initialize the name and the 2 pawns of the player

2. `public boolean isMyTurn();` (Observer)

check if it's players turn

3. `public void setCanPlay(boolean canPlay);` (Transformer)
Set the players turn

4. public boolean hasFinished(); (Observer)

This method check's if the player has finished the game

5. public String toString();
This method creates the string representation for the name of the player

6. public Color getColor(); (Accessor)
gets the color of the player

7. public void setColor(Color color); (Transformer)

set the color of the player

8. public String getName(); (Accessor)

return the name of the player

9. public void setName(String name); (Transformer)

Set the name of the player

10. public Pawn getPawn1(); (Accessor)

get the first pawn of the player

11. public void setPawn1(Pawn pawn1); (Transformer)

set the first pawn of the player

12. public Pawn getPawn2(); (Accessor)

Get the second pawn of the player

13. public void setPawn2(Pawn pawn2); (Transformer)

set the second pawn of the player

Class Deck

Variables:

```
ArrayList<Square> Squares;//array to store the squares
ArrayList<Card> gameCards;//array to store the cards
```

Methods:

1. public Deck(); (Constructor)

Initializing the table

2. public void setTurn(); (Transformer)

This method calculates who will start the game

3. public void getNewCard(); (Accessor)

This method is used for allowing the player do draw a new card

4. public boolean CardsEmpty(); (Observer)

This method is used to check if there are cards left

5. public boolean canFold(); (Observer)

This method is used for checking if the player can fold

6. public void init_Table();

This method initialize the Table

7. public void init_Cards();

This method initialize the array of cards

8. public ArrayList<Card> ShuffleDeck(ArrayList<Card> gameCards);

This method Shuffle the array with the cards

9. public void init_Players();

This method initialize the 2 players

10. public void init_Squares();

This method initialize the squares of the board

11. public void init_Pawns();

This method initialize the pawns of each player

Package Controller

This package contains one class controller. This class is responsible for “playing the game”. It creates the two players,the pawns, the board and the cards. It takes as input the choices of the user from the graphical interface and calls the proper methods to execute the game and update the graphics.

Class Controller

Variables:

```
private final Deck deck;

private final View view;

private Player player1;// player 1

private Player player2;// player 2

int turn=1;// 1 player 1 is playing , 2 player 2 is
playing

boolean cardHasBeenDrawn = false;//This variable informs
```

as if the   new card has been drawn

boolean Fold=false;//This variable infroms us if the fold button   can be activated

//These variables informs as which pawns has been pressed by the   player
boolean red1_pressed=false;
boolean red2_pressed=false;
boolean yel1_pressed=false;
boolean yel2_pressed=false;

private boolean canChangeCard=true;//This variable informs us if we   can allow user to change the card

private boolean Fold_clicked=true;//This variables checks if we can   press the fold button

private boolean move_succeded=false;//This variables informs us if   the move for the pawn has been succeded

Methods:

public Deck getDeck() (Accessor)
return deck of the game

public Controller() (Constructor)
calls the methods for initalizing the game and the graphics

1. public void makeMoves(Card card, Deck deck, int choice, Pawn pawn, Pawn enemyPawn);

This method will call the move methods and the update pawn

public void changeTurn()
This method change the turn when the move has been done or the fold button has been pressed

public void ChangeTurnByMe(int turn)
This method is used to change the round as programmer wants

public void moveToTheEndOfTheSlide(ArrayList<Square> squares,Pawn pawn)

This method moves the pawn that is on start slide square to the end slide square

2. public String whoseTurn();

This method gives us the name of the player that is his turn

3. public void getTurn(); (Accessor)

This method return whose turn is

4. public void init_Game();

This method will initialize the cards , squares , table ,pawns and will set the listener for the card ,mouse,and fold button

5. public void setListeners();

This method will set the listener for the cards , Fold button and pawns

7. public boolean foundWinner()

This method will check if there is a winner

Controller also has 3 inner classes : FoldButtonListener, PawnListener,CardListener

FoldButtonListener :checks if the user can press fold

PawnListener: calls the move pawn methods for the pawn that was clicked , checks if there's a winner and if the move was succesfull , so the next player can draw next card.

CardListener: when new cards button is clicked , updates the card and change the turn

Package View

This package contains the class view. This class create a frame and a panel in it. The panel is separated to two rectangles one on the left half and one on the right half. On the right rectangle there is 2 buttons for the cards , one dialog box for displaying info about the game and one button for the fold. On the left half there is the board which is made by 74 labels which are the squares and the four buttons for the pawns. Also there is a dialog box which will be shown when a player have to make a move.

Class View

Variables:

```
static View mainPanel = new View();// The main panel
static JFrame frame = new JFrame("Sorry Game");// The frame
static JLabel positions[] = new JLabel[72];// position of each
squares
static JButton cardButton = new JButton("sorry card");// button for
the card1
static JButton cardButton2 = new JButton("card");// button for the
second card
static JButton pawn1_red = new JButton();
static JButton pawn2_red = new JButton();
static JButton pawn1_yel = new JButton();
static JButton pawn2_yel = new JButton();
static JLabel start_yellow;
static JLabel start_red;
static JTextArea infoBox = new JTextArea();
static JButton fold_button = new JButton();
```

Methods:

1. `public static void updatePawn(Pawn pawn,Pawn pawn2,Pawn pawn3,Pawn pawn4)`

 This method updates the pawns position on board

2. `public void winMessage(String msg);`

This method display a win message

3. `public void updateInfoBox(String msg);`

This method will update the info box with informations about whose round is

5. public static void init_Components();

This method creates the components for our graphics , images, buttons, text

6. static String askUser(String[] choices);

This method displays a option panel ,which giving the user the options for his next move

7. public static void init_Squares();

This method places the squares to the panel

public int[] ChooseNumOfBlocks()
This method dispays 2 text fields where the user can insert how much blocks he wants to move each pawn when number seven card is activated

public static void updateCard(Card card)
update the card that the player draw

public JButton getFold_button() (Accessor)
get the fold button

public static JButton getCardButton() (Accessor)
get the card button

public static JButton getPawn1_red() (Accessor)

get the first red pawn button

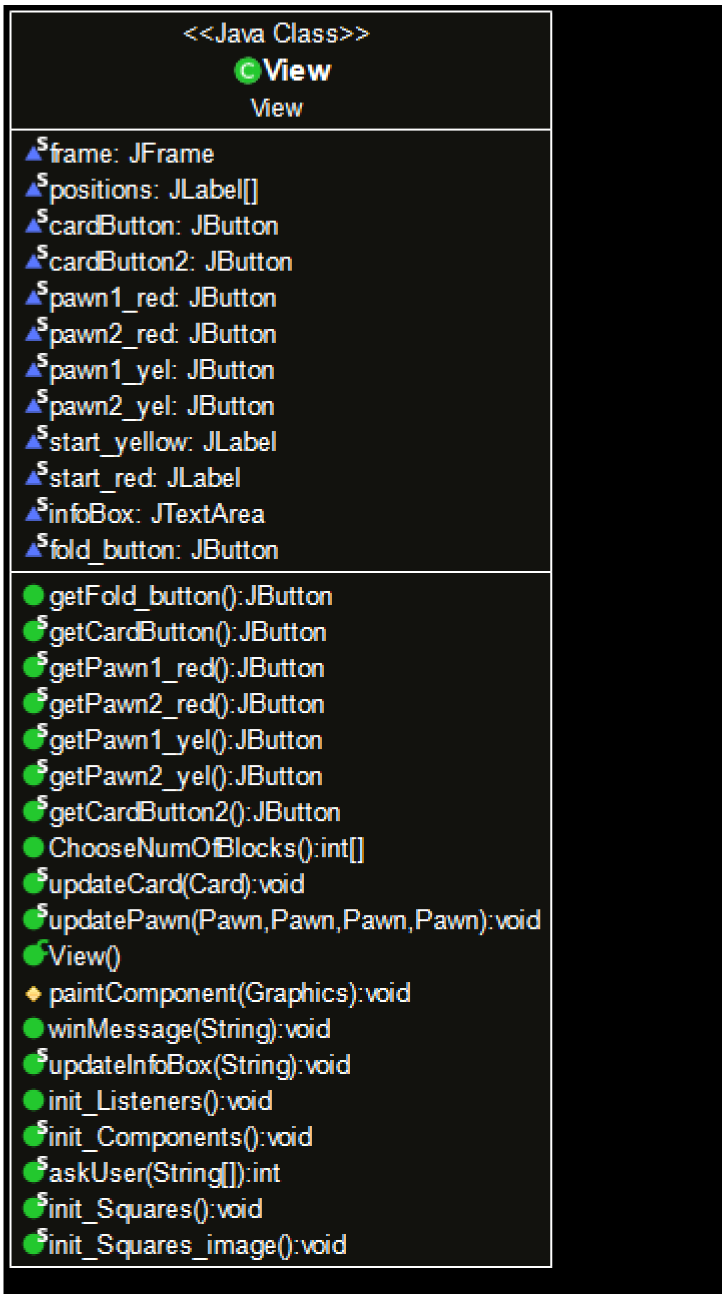
public static JButton getPawn2_red() (Accessor)
get the second red pawn button

public static JButton getPawn1_yel() (Accessor)
get the first yellow pawn button

public static JButton getPawn2_yel() (Accessor)
get the second yellow pawn button

public static JButton getCardButton2() (Accessor)
get the second card's button

UML for View


```
<<Java Class>>
View
View

frame: JFrame
positions: JLabel[]
cardButton: JButton
cardButton2: JButton
pawn1_red: JButton
pawn2_red: JButton
pawn1_yel: JButton
pawn2_yel: JButton
start_yellow: JLabel
start_red: JLabel
infoBox: JTextArea
fold_button: JButton

getFold_button():JButton
getCardButton():JButton
getPawn1_red():JButton
getPawn2_red():JButton
getPawn1_yel():JButton
getPawn2_yel():JButton
getCardButton2():JButton
ChooseNumOfBlocks():int[]
updateCard(Card):void
updatePawn(Pawn,Pawn,Pawn,Pawn):void
View()
paintComponent(Graphics):void
winMessage(String):void
updateInfoBox(String):void
init_Listeners():void
init_Components():void
askUser(String[]):int
init_Squares():void
init_Squares_image():void
```

<u>THE END</u>